Master's Degree in Computer Science

Academic year 2023/2024

RoomFinder

A report for the Mobile Programming and
Multimedia Project

Authors	Student ID
Michael Amista'	2122865
Sebastiano Sanson	2130917

Index of contents

Disclaimer

Note on Gender Language

This document uses the masculine pronoun ("he") to refer to people of all genders. This is done for readability purposes only, and to avoid the constant use of "he/she" or other gender-neutral constructions. Please be assured that the content applies equally to all genders.

Users credentials to test the app

Type	Email	Password
Student	student@gmail.com	123456
Host	host@gmail.com	123456

1 The project

This project has been developed for the Mobile Programming and Multimedia course and it consists in the development of a mobile application using a cross-platform development approach to optimize the writing of the code for both Android and iOS platforms.
This section presents the idea of the mobile app, considering the contribute it can bring through people, and the kind of users who can retain the developed app a useful and effective tool for their daily life.

1.1 Mobile app

Students who want to move to another city for study reasons should have the possibility to examine in a simple way all the rental proposals of the city without making a lot of calls or passing through a lot of rental agencies. This represents something that is missing in the largest Italian cities and that is the reason behind the idea of the chosen application.

The original idea of **RoomFinder** was to develop an app that allows the Italian student community, composed by both national and international students, to easily review all the rental proposals of the various accommodations scattered around the city of studies. The user can examine the various proposals by comparing prices, location and using all the other metrics offered by the app and useful for a particular facility of interest. A further important aspect is the possibility of being able to consult the possible roommates already present through a short description which will inform the user, who is carrying out the search, about the kind of persons he will meet choosing that facility. It is also possible to engage in a chat with the host of the facility of interest with the aim of exchanging useful information for both parties.

1.2 Users segment

The application represents a useful tool for multiple kinds of users. In fact, RoomFinder is not only a useful tool for students who are looking for an accommodation but even for private hosts who want to make available their spaces.

In RoomFinder there are three types of users:

1. **Nonregistered users**: visitors that are not subscribed to RoomFinder. This kind of user can view all the rental proposals present in the system but to access to further features the subscription is necessary.

2. **Registered users**: registered students who are looking for an available and affordable facility. This kind of user can view all the rental proposals present in the system and he can access to further features that can power up his/her user experience. Some of these features are the possibility to save ads, chat with a host to request more information and view the roommates' details for a facility of interest.

3. **Registered Hosts**: registered owners of facilities present in the system. This kind of user can add new ads in the system and/or manages ads already added. Hosts can also respond to requests of information from students via a chat system.

2 Mobile Design

The main RoomFinder pages have been designed using Figma tool. That tool allowed us to easily design pages, considering the mobile design principles seen during lessons, rather than directly code the pages.

The mockups and all the mobile design choices are discussed in this section.

2.1 Mockups

Before starting to design the mockups we had to decide the color palette for the application. Considering that colour theory is a complex theme that interacts also with emotional design and cultures, we decided to make deeper research before choosing the main color and its variations for the palette. What we have found is that **color blue** conveys trust, calmness, professionalism, and inclusivity, all of which are crucial for an app aimed at helping students find reliable accommodations in a new city. The emotional comfort and broad appeal of blue can significantly enhance user experience, making the app a trusted and preferred tool for our users segment. This guided us to the choice of a blue palette for the entire application.

After that we produced different mockups for the main pages of RoomFinder, dividing them based on the type of user who is using the app. All the pages respect the just-in-time principle, showing only the elements that are necessary for that page. Furthermore, all the pages try to reduce the number of user interactions without having crowded layouts and respecting the "content always on top" principle.

2.1.1 Nonregistered user

FIRST VISIT

This page is the first one every user will see the first time he runs RoomFinder. The idea is to transmit a message to capture the user curiosity through the **humanization**, involving the user through a humanized figure that represents happiness, calmness and trust of students like him that found their stability using RoomFinder.

Furthermore, there is a written message that encourages students and hosts to use and be part of the RoomFinder network.

At the end the user, if he is convinced, is ready to start the experience through a tap on "*Let's start*" button which redirects the user to the rental proposals search page, the homepage of RoomFinder.
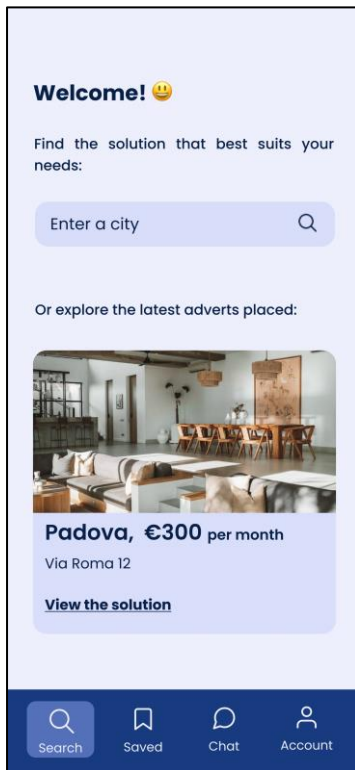
RENTAL PROPOSALS SEARCH

After tapping on "*Let's start*" button, the user lands on the rental proposals page, the homepage of RoomFinder, where he can search offers for a particular city of interests or explore some of the latest inserted adverts. The layout of the page is simple and gives to the user all the information and instructions it needs to begin the experience in RoomFinder. The above part of the page allows users to search for facilities located in the digited city and the rest of the page is dedicated to show to users some of the latest ads. Users can view more details for a single insertion by tapping anywhere inside the facility box or more intuitively by tapping on "*View the solution*".

The content of the page is easily reachable reducing the user effort. The bottom bar provides the access to all other main pages of the application, even if some of them are available only for registered users (as saved ads, chat or account pages). Even if some of the pages are reserved to only registered users, the idea is to show to the user all he can do in the application and then, when he tries to use a functionality that requires a RoomFinder account, redirect him to the login page (discussed later in the document) explaining him why.
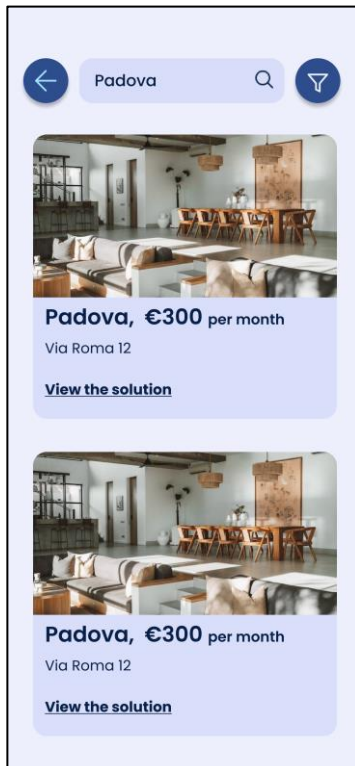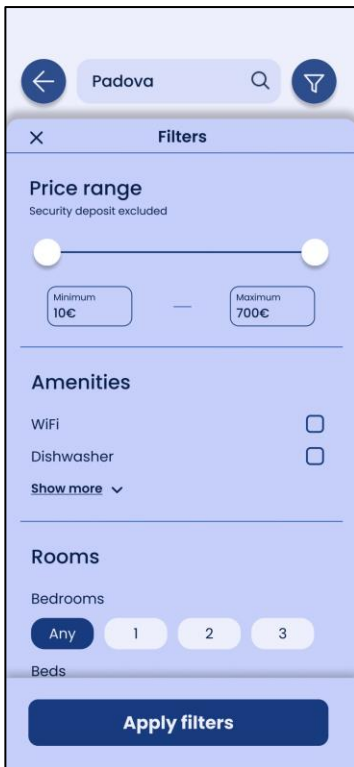
SEARCH RESULTS

This page contains the results, so all the rental proposals, for the city searched by the user through the search functionality present in the previous page (rental proposals search). The results are presented as a list of items of the same form seen for the previous page and so with the same already discussed modalities to view them.

To decrease the number of user interactions the search bar has been placed even in this page for further user searches. In this way, if the user wants to search the proposals of another city, he can simply enter the new city in the search bar placed in the top area of the page to avoid wrong taps.

It is also possible to close the page, returning so to the rental proposals search page, by using the back button placed on the left corner, again to avoid wrong taps by users.

Therefore, since the results for the searched city can be a lot, the user can filter them using a filter menu that appears through a tap on the top right icon.

SEARCH RESULTS – FILTER MENU

As mentioned before, the filter menu allows users to filter the search results to match their preferences for the different proposals in the system.

This menu contains several voices that can be used to search the rental proposals that match what user is looking for. At the end of the panel there is a confirmation floating trigger button to search for the customized results.

The panel can be closed in three ways, by tapping on the top left "*X*", tapping outside the panel itself or using a flick gesture with a movement that starts from the top area (approximately from the "*Filters*" voice) to the bottom of the screen.
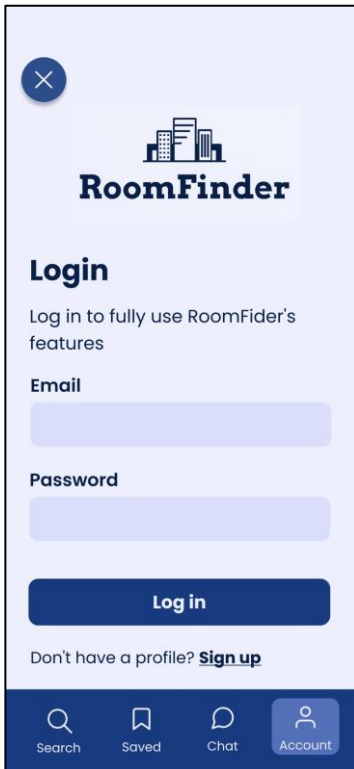
DETAILED PAGE FOR A FACILITY

This page represents a detailed page for a facility present in the system and on which user tapped during its navigation. The page contains all the information needed to better understand what the facility offers.

In the top area we have a slider which shows some photos of the room. In general, it is true that sliders should be avoided if possible because they cause the loss of the user's overall vision. In this case the user has full control over the slider, without the problem of annoying timers that update continuously the slide. Furthermore, the user can expect what comes next since it a simple slider of the facility photo. Users can move on and back the slides with a simple natural gesture from the left or right depending on the movement they want to do. The gesture is natural and for this for us it is not necessary to teach it. Furthermore, in every moment user knows at which point of the slider he is thanks to the numbered label on the right side. In the top area there are also two other buttons, a back button on the left and another button on the right to save the insertion.

In the page there is even the possibility to open a detailed page about the current renters already present in the facility, but this is a functionality available only for registered users, so when a user taps on "*More details*", as before, he will be redirected to the login page. At the end it is possible to request further information, regarding the facility, directly sending a message to the host through the floating trigger button "*Request information from the host*" placed in the bottom part of the screen.
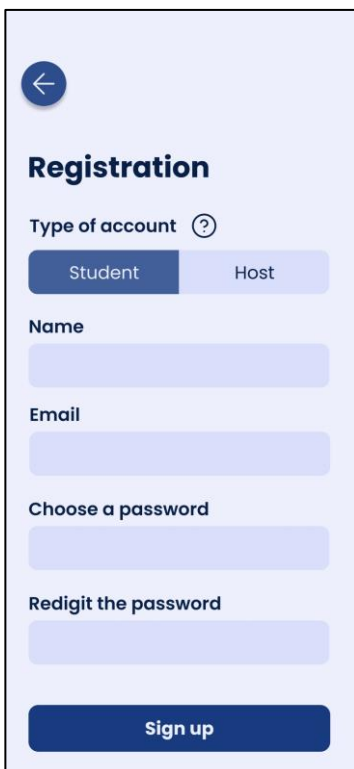
LOGIN

This page offers the possibility to access to the RoomFinder account. Users can be also redirected to this page when they try to use a functionality not available for non-authenticated users. In this case, since the user is automatically redirected here, the page provides an explanation about the necessity to have an account to use all the functionalities that RoomFinder offers, ideally including the one that redirected the user here.

The page is simple, and it provides a standard form to authenticate users by an email and a password. The labels for both inputs have been placed above the input to avoid that users hide them unintentionally during digiting the input.

If the user has not a profile it is possible to access to the registration page tapping on "*Sign up*" voice.

To avoid mistake by the user in the comfort zone, the "*cancel*" button, useful to close the login page and to return where the user was, has been placed in the top left corner.
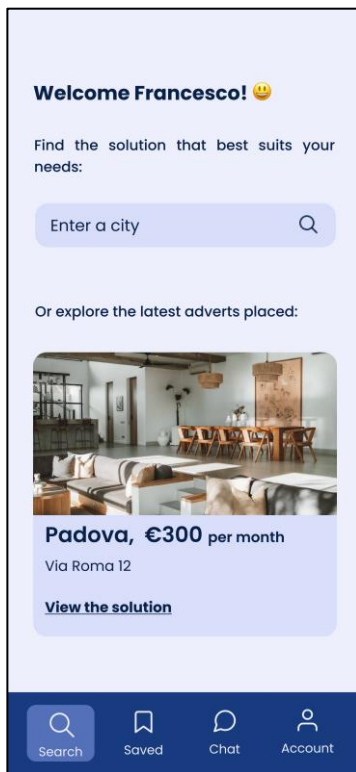
REGISTRATION

To create a new account every user can pass through the registration page. The layout of this page is pretty much the same of the login one, so all the considerations discussed for the login page are still valid here.

The only thing that changes is the form field "*Type of account*" which is related to the purpose for which the user is creating a new account, so to be a student who is searching for a house or a host who want to publish ads. The user has to select one of the two kinds of account by tapping on the label "*Student*" or "*Host*". To make everything clear an info button, placed near that field, allows to obtain more information about the different types of account, ideally opening a pop-up that can be closed after everything is clear.

Talking again about the "*Type of account*" button, there is a preselected option which is highlighted with a darker color, different from all the other uncomplete form fields in the page, to indicate the chosen option. This is quite intuitive because the difference of color compared to the other fields in the page can suggest which is the current chosen option. Furthermore, this implementation choice is even explained tapping on the info button, previously discussed.

All the above pages are available for both registered and nonregistered users so, if those pages do not present layout or content changes depending on the type of user who is using the app, they will not be discussed anymore to avoid redundant explanations.

2.1.2 Registered user (Student)

RENTAL PROPOSALS SEARCH

The rental proposals search page is identical to the counterpart available for nonregistered users, so all the considerations made for that page are still valid for this version. The only element that changes in the content is the "*Welcome user_name!*" message.

Even the bottom menu bar is the same with the only difference that now, since the user logged in, all the RoomFinder pages and functionalities are available.
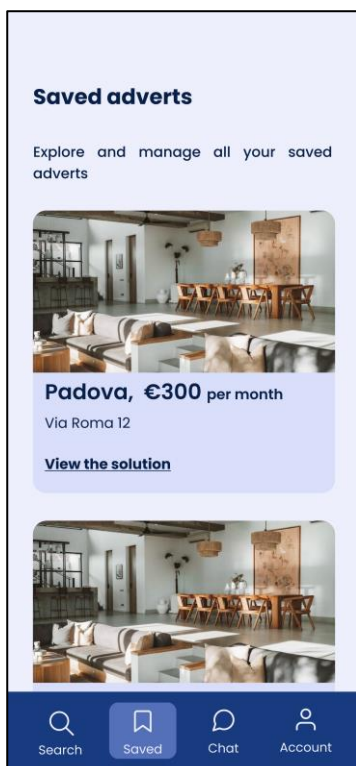
SAVED ADVERTS

This page contains all the rental proposals saved by the user during its navigation. The proposals can be saved by tapping on the save icon present on the specific rental proposal detailed page. Users can also remove the saved proposals by tapping on the same icon used to initially save them. The icon changes its look depending on if it can be saved or it has been already saved by replacing a void bookmark icon with a filled bookmark icon respectively. For us this approach is natural and effective, thinking also on what happens in social media apps, like Instagram, when the user put or remove a like on a post.

Since the appearance of the results list is the standard one used among all RoomFinder pages, the single saved proposal can be opened, with same modalities already discussed, to view the detailed page, again by tapping on the whole box of a single proposal or, more intuitively, tapping on the "*View the solution*" voice.
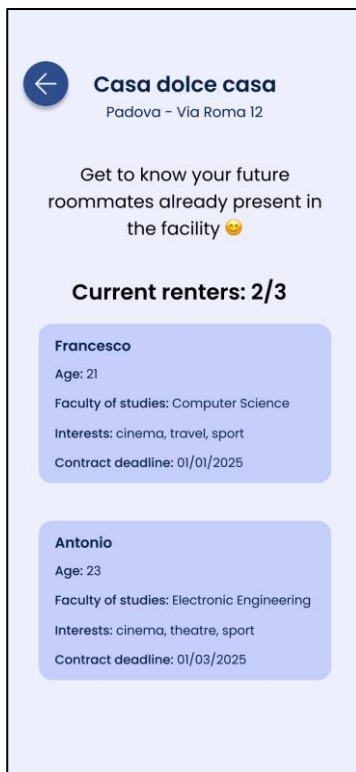
CURRENT RENTERS

In the detailed page for a facility, it is possible to get more information about the current renters. Only registered users are allowed to open this page.

The content is simple, and it provides a way to know better the future roommates with the user will live if he chooses that facility. Every roommate is represented by a box containing some general information about him.

It is possible to go back to detailed page of the relative facility by tapping on the left button, near which a general summary of the facility has been placed to remember user where it was before landing on this page.
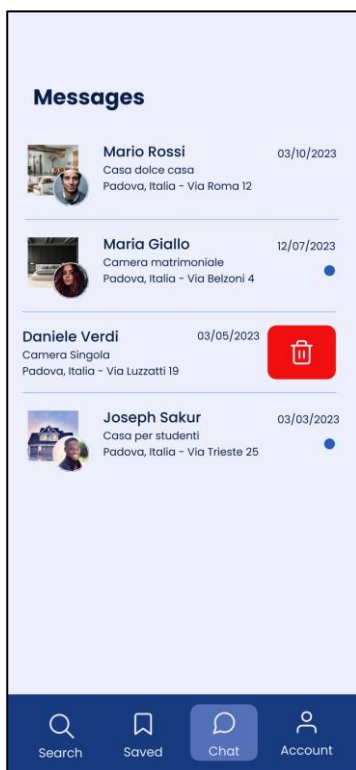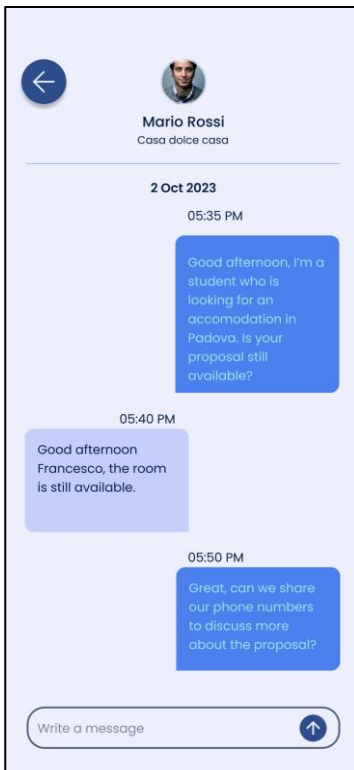
CHAT

This page allows to manage all the messages to hosts. Messages are grouped in different chats, each one representing the request made for a specific rental proposal. New messages are highlighted by a blue circle on the right area of each chat.

It is even possible to delete old chats with a flick gesture on the specific element, starting the movement from the right to the left area of the chat itself. This gesture does not require high precision and we think that after a while it can become a sort of reflex for the user, decreasing so the required effort. This is due also to the fact that similar gestures are already used in other common application (for instance Gmail) to delete messages, so this implementation is really similar to something that users could have already seen or experimented in other scenarios. This gesture will be taught the first time the user lands on the chat page.

Then, tapping on one chat, more precisely on the whole horizontal area occupied by each element, it is possible to view every message sent to request information about a specific rental proposal.

DETAILED PAGE FOR A CHAT

The page represents the whole history of messages sent to a host regarding a facility of interest. The layout of the page is really similar to every chat application present in the market. On the right it is possible to find the student messages and on the left the host ones. It is possible to write and send a new message using the input text field on the bottom of the page.

In the top part of the page there is a short summary of the rental proposal for which the student is writing.

It is possible to go back to the chat page using the top left button.
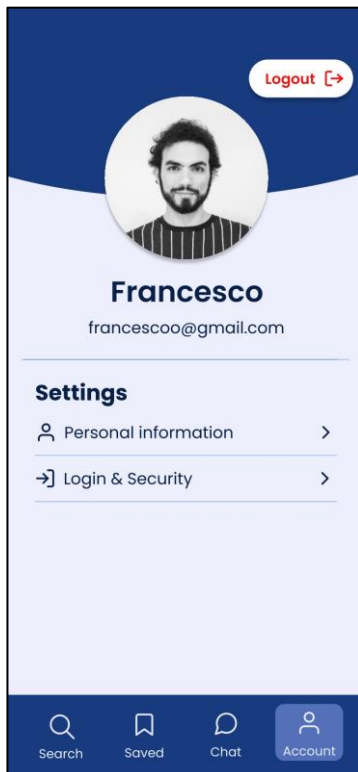
ACCOUNT

This page allows users to view and manage their profile information. It is also possible to customize the app experience by some general settings, as discussed before for the counterpart available for nonregistered users. It is also possible to logout using the top right button which has been placed outside the comfort zone to avoid mistakes by user even if a confirmation dialog to complete the operation is however shown.

2.1.3 Registered user (Host)
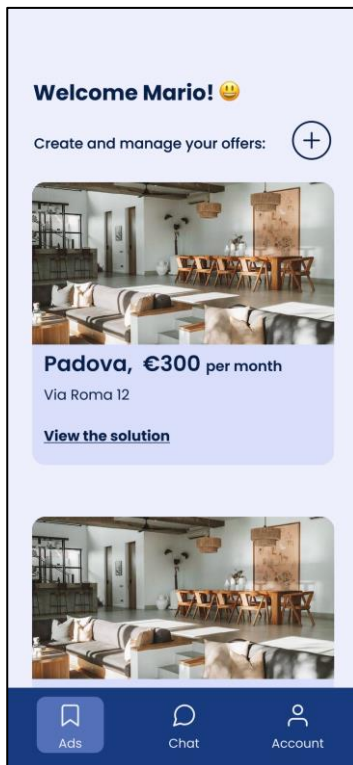
RENTAL PROPOSALS MANAGEMENT

This page represents the first page that a registered host sees when he starts RoomFinder. This page allows to create and manage rental proposals that can be viewed by students who want to search for an accommodation.

The overall layout is really similar to other pages previously discussed. The host can view the whole list of his/her proposals and can add new ones by tapping on the "+" button placed on the top right area. The user can also view in detail each proposal that appears in the list tapping on the whole item box or, more intuitively, tapping on the label "*View the solution*".

DETAILED PAGE FOR A FACILITY

The detailed page is pretty identical to the counterpart available for other kinds of user with just a few differences. The host has two more features represented by the two buttons placed in the top right area which allow respectively to edit the proposal or to delete the proposal. Since both buttons allow to modify the page content, they have been placed outside the comfort zone to avoid wrong taps by user. Anyway, even in this case, the delete option opens a confirmation dialog that requires the user confirmation before proceeding with the elimination.

Since the chat and the account pages are the identical to the counterparts available for a registered student, they are not discussed again to avoid redundant explanations.

PAGES FOR ADD/EDIT RENTAL PROPOSALS

These standardized screens, employed throughout the rental proposal creation/modification process, feature consistent navigational elements:

- **Back Button**: allows the user to return to the previous step in the workflow, it is positioned in the top-left corner of the screens, outside the comfort-zone;
- **Cancel Button**: allows the user to cancel the add/edit action at any time, it is positioned in the top-right corner of the screens, except for the first wizard screen where it is placed in the top-left corner, outside the comfort-zone;
- **Info Button**: represented by the "?" icon placed near each page title. It provides useful information to better understand the function of the page;
- **Next Button**: enables progression to the subsequent stage;
- **Progress Indicator**: visually communicates the user's current position within the multi-step process, mitigating potential disorientation.

These persistent components enhance usability by providing a clear, intuitive path through the creation/modification journey. Every page is also characterized by a help button, identified by the symbol "?", to obtain more information about the page itself.
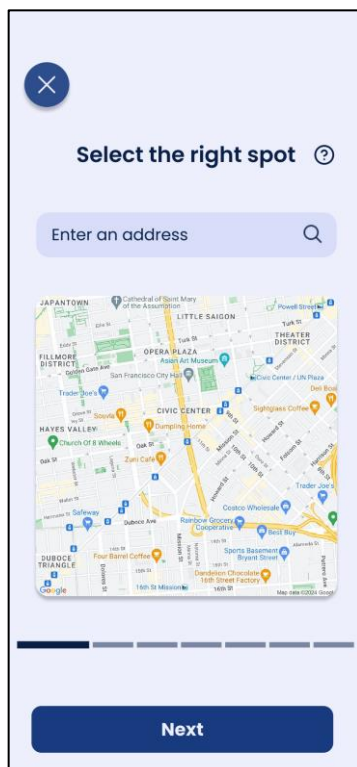
ADD/EDIT RENTAL PROPOSAL – PAGE 1

This initial screen in the rental proposal creation/modification workflow allows users to pinpoint the accommodation's address. This can be achieved by either:

- **Direct Input**: Manually typing the address into the search bar.
- **Map Interaction**: Navigating and selecting the location on the embedded map.

The initial screen in the process deviates slightly, incorporating a "Cancel" button that allows the user to terminate the wizard workflow entirely. This provides an emergency exit, ensuring users are not locked into the process if they wish to abandon their changes.

ADD/EDIT RENTAL PROPOSAL – PAGE 2

The "Set the Rooms" screen, appearing as the second step in the workflow, streamlines the process of defining the accommodation's capacity. Users can easily indicate the quantity of bedrooms, beds, and bathrooms using intuitive plus/minus buttons, which increment or decrement the values for each category.

ADD/EDIT RENTAL PROPOSAL – PAGE 3

The "Manage Renters" screen serves multiple purposes:

- **Maximum Guests**: A field allows users to specify the total number of guests allowed for the accommodation.
- **Add Renter**: Strategically positioned above the renter list, a "+" button provides a convenient way to add new renters to the proposal. This placement eliminates the need for users to scroll down the list to add entries. Basic renter information is collected at this stage, with more comprehensive details captured on the subsequent screen.
- **Edit/Remove Renter**: Each renter entry in the list includes options to either edit the associated information or remove the renter entirely. This inline functionality streamlines the modification process and eliminates the need for users to restart

ADD/EDIT RENTAL PROPOSAL – PAGE 3 (ADDING/EDIT RENTERS PANEL)

The "Add/Edit Renter" popup, triggered by the corresponding buttons on the "Manage Renters" screen, emerges from the button's location to ensure optimal interaction. This popup facilitates the input of detailed renter information, comprising:

- **Name**: The renter's full name.
- **Faculty of studies**: The renter's faculty of studies.
- **Interests**: The renter's interests.
- **Contract deadline**: The renter's contract deadline.

By originating from the button itself, the popup minimizes the need for users to shift their attention or make extensive movements on the screen. This strategic design fosters a seamless and efficient workflow when adding or modifying renter details.

Furthermore, the "Contract Deadline" field, providing flexibility in date entry:

- **Manual Input**: Users can directly type the contract deadline date into the field.
- **Calendar Widget**: Clicking on the field opens a calendar widget, allowing users to visually select the desired date.

This dual approach caters to different user preferences, accommodating both those who prefer manual entry and those who find a visual calendar more intuitive.

ADD/EDIT RENTAL PROPOSAL – PAGE 4

The "Amenities and Services" screen empowers users to curate the accommodation's offerings:

- **Amenity Selection**: Users can choose from a predefined list of amenities to indicate those available at the property.
- **Add Service**: Conveniently located above the amenities list, the "Add Service" button facilitates the creation of custom services not included in the standard options. This strategic placement eliminates unnecessary scrolling for users.

By combining pre-populated choices with the option to add personalized services, this screen ensures a comprehensive representation of the accommodation's features and offerings.

ADD/EDIT RENTAL PROPOSAL – PAGE 5

The "Set the monthly rent" screen simplifies the process of defining the accommodation's rental cost. A slider component provides an intuitive mechanism for users to select the desired monthly rent amount. The slider's range and incremental values allow for precise adjustments, ensuring accurate representation of the rental price.

ADD/EDIT RENTAL PROPOSAL – PAGE 6

The "Let's give your house title" screen presents a text field where users can input a descriptive and engaging title for their rental listing. This title plays a crucial role in attracting potential renters and summarizing the key features of the accommodation.

ADD/EDIT RENTAL PROPOSAL – PAGE 7

The "Choose Photos" screen facilitates the selection and management of accommodation visuals:

- **Add Photo Button:** Strategically positioned above the photo grid, this button enables users to easily add new images to the listing without needing to scroll.
- **Photo Grid:** Uploaded photos are displayed in a grid format, each marked with a number indicating their display order.
- **Remove Photo:** A dedicated button on the top-left corner of each photo allows for individual photo removal, offering flexibility in image curation.
- **Review Listing:** Unlike previous screens, this screen features a "Review Listing" button at the bottom, providing direct access to the final review stage for a comprehensive overview before submission.

ADD/EDIT RENTAL PROPOSAL – PAGE 8

The final screen provides a comprehensive overview of the accommodation details entered throughout the wizard process. This step allows users to verify the accuracy of the information, including the address, room configuration, guest capacity, amenities, monthly rent, listing title, and uploaded photos. This final check ensures that the listing is complete and ready for publication, minimizing the risk of errors or omissions.

3 Flutter framework

RoomFinder has been developed using the cross-platform Flutter framework. This framework has been chosen for its high popularity and for the native look and feel of its outputs.
This section presents the framework, its classification according to Raj and Toleti and the different advantages that it provides.

3.1 Framework

Flutter is an open-source UI software development toolkit created by Google. It is primarily used for building natively compiled applications for mobile, web, and desktop from a single codebase. Released in 2015, Flutter has rapidly gained popularity among developers due to its efficient development process and high-performance rendering capabilities. The framework utilizes the Dart programming language and includes a rich set of pre-designed widgets, making it easy to create visually attractive applications with a seamless user experience.

3.2 Classification according to Raj and Toleti

According to Raj and Toleti cross-platform frameworks classification, Flutter follows a **cross-compiled approach**. This ensures that Flutter apps can achieve high performance similar to native applications while maintaining a single codebase. The cross-compilation approach used by Flutter leverages the Dart language's Ahead-Of-Time (AOT) compilation to produce fast and predictable native code, which is a key characteristic of cross-compiled frameworks according to Raj and Toleti.

3.3 Advantages

One of the major advantages of Flutter is its ability to enable **cross-platform development**, allowing developers to write code once and deploy it across multiple platforms, such as Android, iOS, web, and desktop. This significantly reduces development time and costs. Additionally, Flutter's **hot-reload** feature facilitates a faster development cycle by allowing developers to instantly see the results of their changes without restarting the application. The framework also provides **a rich set of customizable widgets**, ensuring a consistent look and feel across different devices and screen sizes. Furthermore, Flutter's **strong community support** and **comprehensive documentation** make it an accessible and powerful tool for both novice and experienced developers.

4 Project Architecture

Before starting any project development choosing the project architecture is a fundamental phase where studies and research are crucial to find the most suitable architectural pattern for the project. In the case of RoomFinder we made different analysis and studies of which architecture could be the most suitable for our application. There are several architectures for Flutter projects, from the ones designed for bigger applications to the ones for smaller applications.

As we will discuss later on this document, since for our project we decided to use the Riverpod external framework to clearly separate the UI from the business logic enhancing performances in reconstructing the widget tree, we needed an architecture highly compatible with Riverpod framework. We made several research before finding a "not official" architecture that is the one we chose, highly compatible with Riverpod framework and that helps maintaining code over the time. This architecture has not an official name, so we are going to call it "Riverpod Architecture" as the writer of the article on which is based calls it. We report here the article: https://codewithandrea.com/articles/flutter-app-architecture-riverpod-introduction/.

A schema of Riverpod Architecture is following reported:
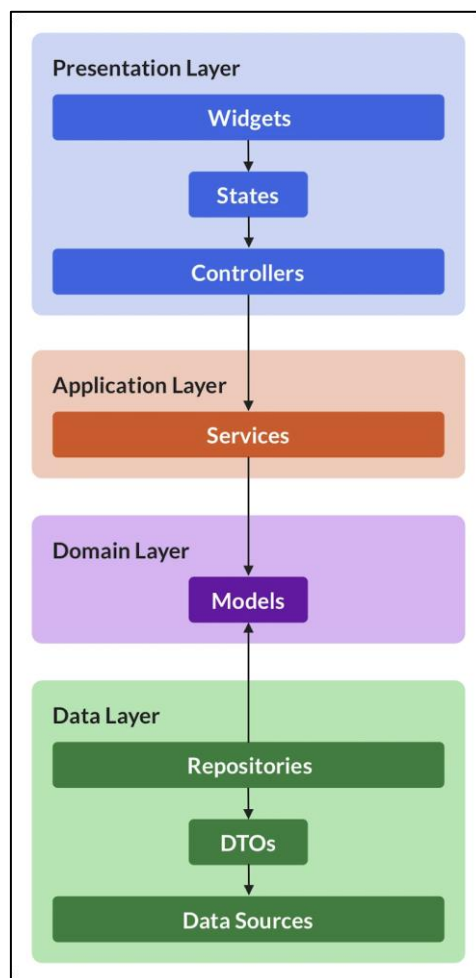
The architecture is formed by four layers that manages all the different aspects of a Flutter app:

1. **Presentation Layer**: responsible for the UI.
2. **Domain Layer**: it defines application-specific model classes that represent the data that comes from the data layer.
3. **Data Layer**: responsible for retrieving data from external services as a database.
4. **Application Layer**: typically used to handle the logic for more complex applications. This layer is completely optional and since it is used to handle a more complex logic compared to the one used in RoomFinder we didn't consider this layer to structure the overall architecture.

Let's analyse better the three layers that we used to implement the Riverpod Architecture in our project.

PRESENTATION LAYER

This layer is formed by three components:

1. **Widgets**
2. **States**: the Flutter native ones and not only. Here they also refer to the states provided by Riverpod and called "Providers". Providers are a sort of global states that can be observed from widget that extends ConsumerWidget and ConsumerStatefulWidget classes, the counterparts respectively for StatelessWidget and StatefulWidget Flutter classes and used to let the widget access to the global state.
3. **Controllers**: they handle asynchronous data changes and widget state. A typical controller operation is to handle user input and then update the widget state which in turn triggers the UI update (e.g. pre and post login action).

The following schema shows the exact workflow inside the presentation layer:
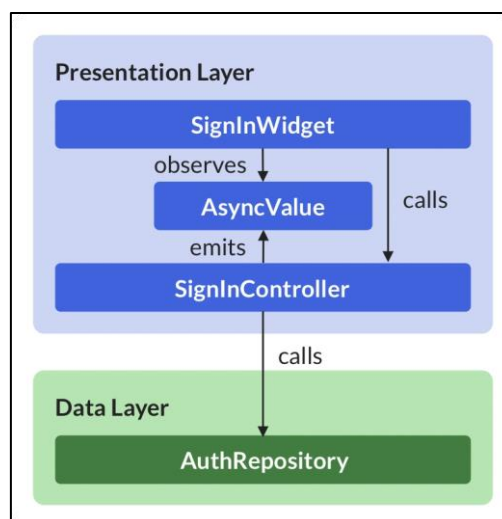
DOMAIN LAYER

It only contains the **Models** classes used to represent the data collected by the data layer.

DATA LAYER

This layer is formed by three components:

1. **Data Sources**: third-party APIs used to communicate with the outside world (e.g. a remote database, a REST API client, a push notification system, a Bluetooth interface).

2. **Data Transfer Objects (DTOs)**: they are returned from data sources. DTOs are often represented as unstructured data (such as JSON) when sending data over the network.

3. **Repositories**: they are used to access DTOs from various sources, such as a backend API, and make them available as model classes to the rest of the app through the domain layer.

5 RoomFinder application

This section presents the final application, the structure of the code folder, all the external services used to make it all work, the testing phase and all the future implementations that can increase the functionalities of the application.

5.1 Final application

This section describes in detail the final application developed using Flutter framework and based on the mockups and design choices made during the mockups realization. All the considerations made in the mockup section are valid also for the final screens explained in this section, due to this the design choices are not explained here.
We divided the app explanation in the three different workflows that the app can assume, based on the type of user who is using RoomFinder, similar to the mobile design section discussed in this document.

5.1.1 Nonregistered user

FIRST VISIT

As previously discussed, this page is the first screen that a new user sees after running RoomFinder for the first time. It is a kind of welcome message that simply tries to explain the goal of this application with the aim of attracting students or hosts that want to be part of RoomFinder mission: make the student rental proposals search easier.

Once the user taps on "*Let's start!*" button he is redirected to RoomFinder homepage where he can start to search and examine the different proposals that RoomFinder makes available, finding the most suitable for his needs.
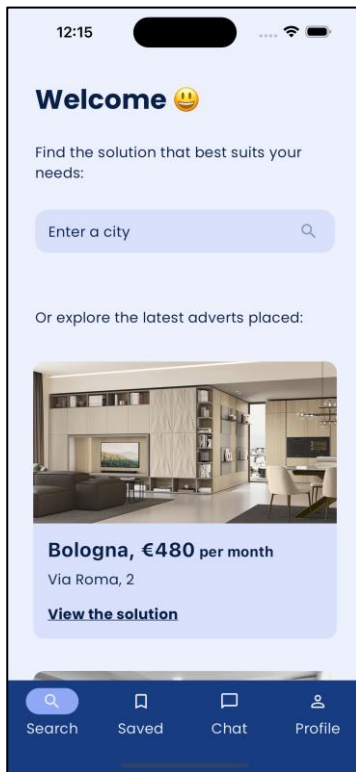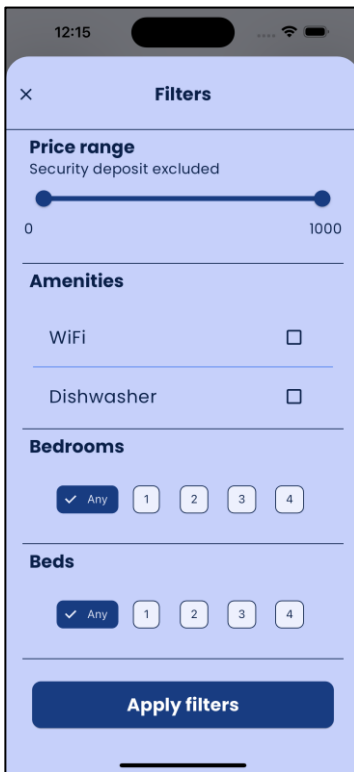
RENTAL PROPOSALS SEARCH

This is the RoomFinder homepage where the user can search rental proposals for a city of interest, or he can view the latest adverts placed for a specific city present in the platform. To avoid the user sees the same adverts every time he lands on this page, the city is chosen randomly in a way to do not show always the same proposals.

As previously discussed, users can view a particular proposal by tapping on the proposal box area or more intuitively on "*View the solution*" label. If the user wants to search proposals for a specific city of interest, he can tap on the search bar, digit the city and then tap on the search button to load the results, if any.
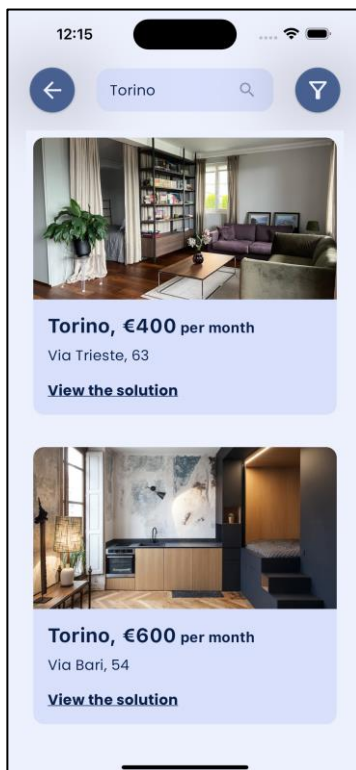
SEARCH RESULTS

This page contains the results for user search. It is possible to come back to the homepage using the back button, search again other proposals by changing directly here the city, decreasing so the number of user interactions (instead of returning to the homepage), or filter the displayed results by tapping on the top-right button that opens a filter panel.

SEARCH RESULTS – FILTER MENU

This filer panel can be used to filter the previously search results by making another database query that removes the rental proposals that do not respect the filter parameters.

Users can filter results by price range, amenities, number of rooms (Bedrooms, Bathrooms), number of beds and number of roommates already present in the facility.

Once the user has selected all the useful filters he can load the filtered search by tapping on "*Apply filters*" floating trigger button. Otherwise, he can close the panel by tapping on the "*x*" icon placed on the top-left corner, tapping outside the panel itself or using a flick gesture with a movement that starts from the top area (approximately from the "*Filters*" voice) to the bottom of the screen.
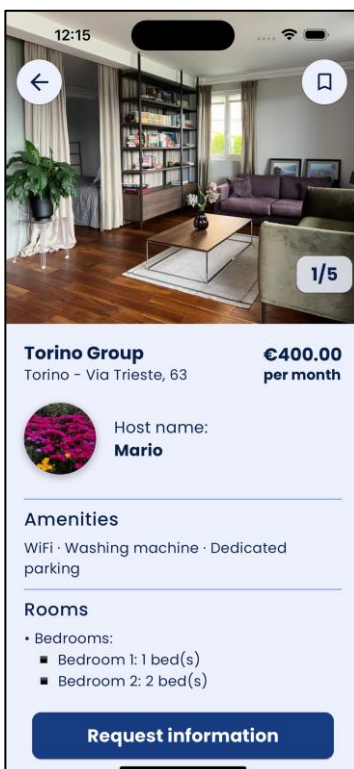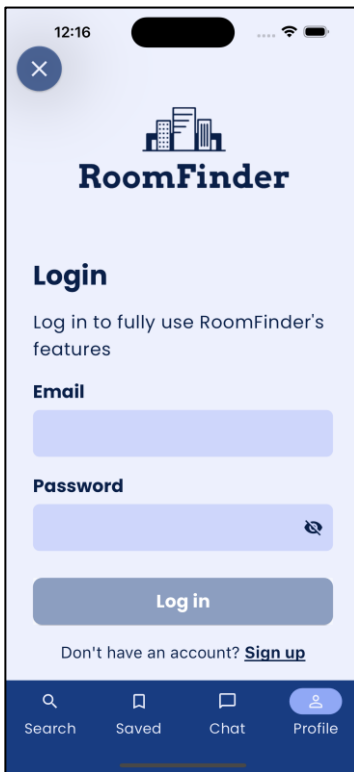
DETAILED PAGE FOR A FACILITY

This page allows users to examine the details of any rental proposals they can find across the app by tapping on rental proposals containers.

The page is simple and respects its mockup. Users can view facility photos, using a carousel with the same modalities already discussed in the mockup detailed page section, and observe any other facility detail.

Also, users can save the proposal by tapping on the top-left button or request information for that facility by tapping on the bottom floating trigger button "*Request information*" which redirects the user to the chat page to "ideally" chat with the host responsible for that facility. We say "ideally" because, as we will discuss later on this document, due to time constraints we were not able to develop the chat functionality which is just "drawn" in the app to show our UI idea for that page.

As discussed in the mockup section, both save and chat functionality are available only for registered users so, if a no logged user tries to use them, he is redirected to the login page that explains with a simple message that to use all the RoomFinder functionalities it is necessary to have an account.
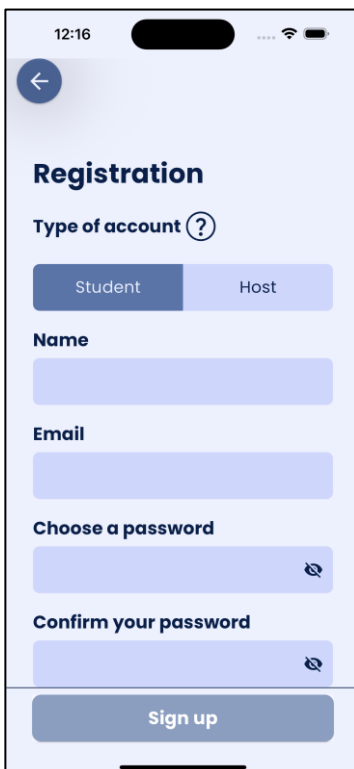
LOGIN

This page contains a simple login form to access to RoomFinder.

Both email and password fields have a client validation when the user taps on "confirm" keyboard button that highlights possible client-side errors. The "*Login*" button is disabled until the client-side errors for both fields are corrected.

If the credentials are wrong a snack bar error message appears after tapping on "*Login*" button telling the user that there is an error in the entered credentials. Otherwise, if the credentials are correct the user is redirected to the homepage with an appropriate welcome message that indicates the login happened correctly.

There is also the possibility, for those that do not already have a RoomFinder account to sign up to the app by tapping on "*Sign up*" label under the "*Login*" button.

If the user does not want to login or sign up to RoomFinder he can simply use the cancel button placed on the top-left corner to return to the homepage.

REGISTRATION

This page contains a simple registration form to sign up to RoomFinder.

Name, email and password fields have a client-side validation when the user taps on "confirm" keyboard button that highlights possible client-side errors (as empty name, not correct email format, passwords do not match, etc).

There is also a "*Type of account*" button that enables the user to create an account with the role of student or host. Users can find, if necessary, all the information in the helper button "?", placed near the "*Type of account*" label, to understand which is the right choice for their use.

"*Sign up*" bottom floating trigger button is disabled until the client-side validation is valid. If there is not any client-side error the user can register to RoomFinder by tapping on "*Sign up*" button. After that, the user is redirected to the login page to test the chosen credentials. Anyway, this is just a way to make the user feel more secure with his login credentials; in fact, if he tries to close the login page the authentication is automatically handled after the "*Sign up*" tap.

If the user wants to simply close this page, he can tap on the top-left button to return to the login one.

5.1.2 Registered user (Student)
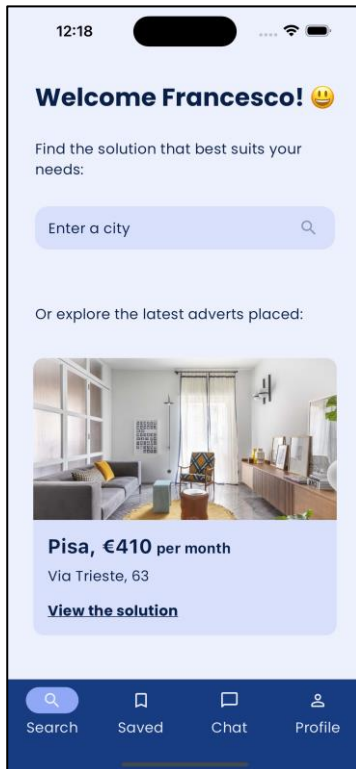
RENTAL PROPOSALS SEARCH

This is the same homepage discussed before for the nonregistered user but with some differences. In fact, the logged user can find a different welcome message with his name and all the forbidden features for nonregistered users are now available and ready for the use.
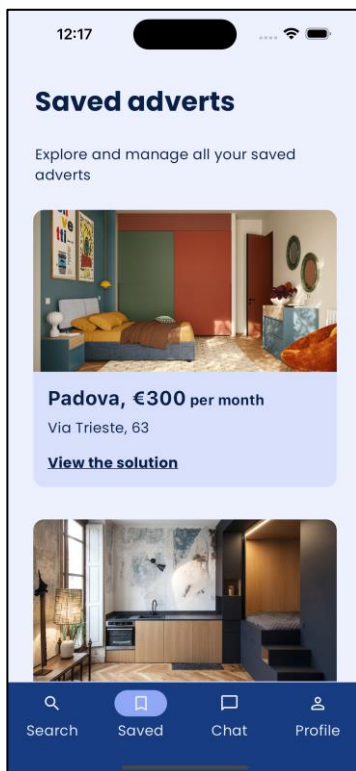
SAVED ADVERTS

This page allows user to handle all his saved rental proposals. To delete a saved adverts the user needs to tap on the specific rental proposal, go to the facility detail page and tap the saved icon which will be initially "filled", indicating the ad is currently saved by the user.

CURRENT RENTERS

Scrolling the facility detail page, it is possible to note a voice "*Current renters: number_of_current_renters/facility_renters_capacity*". If the number of current renters is greater than one, it is possible to access to the current renters page that shows to the user some info regarding the current renters in the facility of interest to match some common interests or just to have an idea of his possible roommates. The current renters page is a feature for only logged users.

The page content is simple and easily readable, each renter has its own box where there are some details regarding his name, age, faculty of studies, interests and contract deadline.

If the user wants to exit from the page, he can use the top-left back button to come back to the facility page.

CHAT

This page represents our mockup idea for the chat page, in fact due to time constraints we were not able to develop the chat functionality and therefore we only "drawn" it.

The chat page contains the list of all chats, so any request of information for a facility of interest from the student point of view. Tapping on each chat it is possible to open the detail page that shows all the messages sent and received by the host.
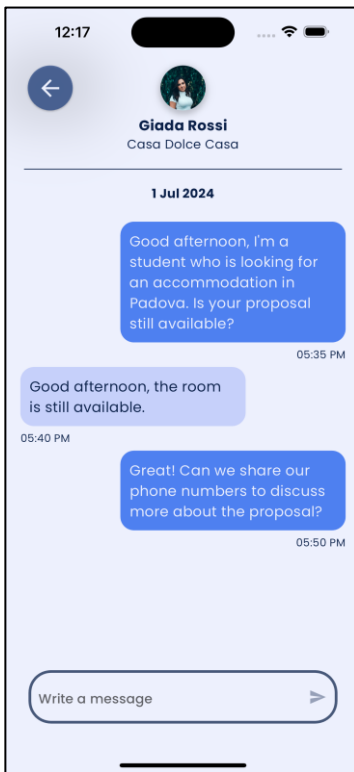
DETAILED PAGE FOR A CHAT

As mentioned before, this is our idea for a chat detail page containing all the messages shared with the host.
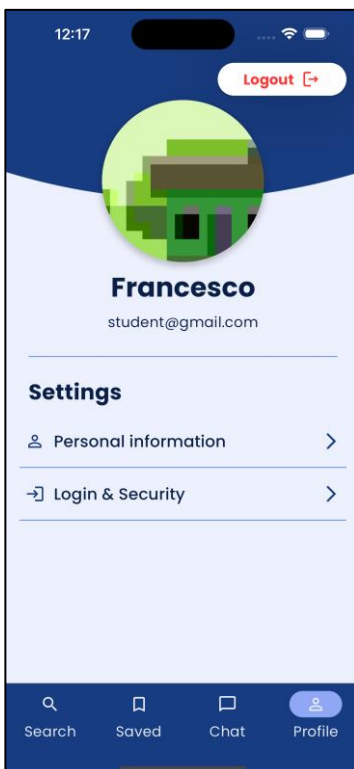
ACCOUNT

This page represents the account page of the current logged user.

User can logout tapping on the top-right button which, when tapped, shows a confirm dialog asking if the user is sure to logout from RoomFinder.

In this page it is also possible to handle the settings for current logged user account such as personal information editing and password change.

PERSONAL INFORMATION

The personal information page allows the user to change the current profile photo (using the camera or the gallery) or his name. If any change is applied the "*Submit change*" button will be available to save the changes and if the user tries to exit from the page with pending changes an alert dialog is shown asking if he is sure to leave the page.

LOGIN & SECURITY

The login and security page allows the user to change his current password. The client-validation is the same used in the registration page, so if the passwords do not match or do not respect the requirements an error is shown on submitting the field with keyboard.

5.1.3 Registered user (Host)
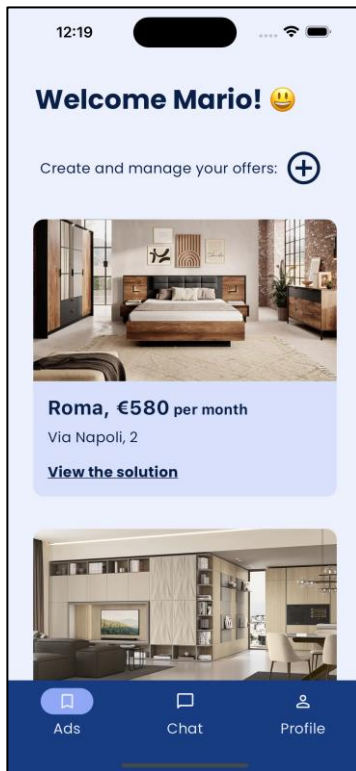
RENTAL PROPOSAL MANAGEMENT

This is the RoomFinder homepage for the registered host where he can view all the proposals he has created.

Like for the registered student, there is a personalized welcome message with his name, then, as for other kinds of user, the host can view in detail a proposal.

There is also a button placed in the top-right corner of the screen that allows the user to start a wizard of creation to add a new proposal.
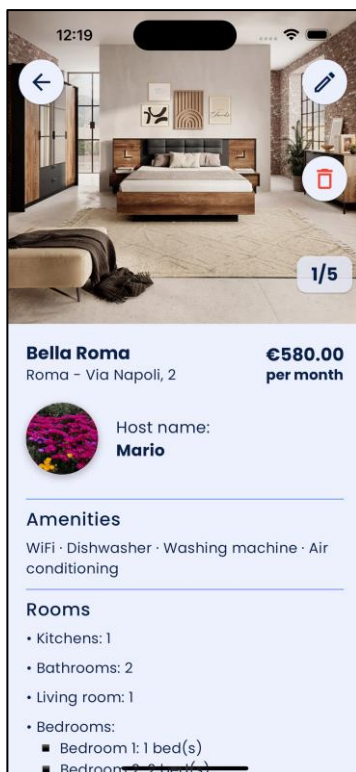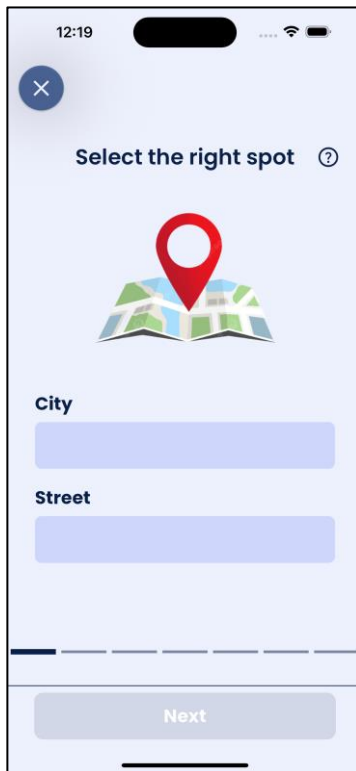

DETAILED PAGE FOR A FACILITY

This page is essentially the same detail page for a facility discussed for the other users, except for some new buttons through which the host is able to delete or edit a proposal, by tapping the corresponding button.

PAGES FOR ADD/EDIT RENTAL PROPOSALS

Once clicked the "add" (in the host homescreen) or "edit" button (in the facility detailed page), the user opens a wizard that drives him through the process of creation/editing of a proposal.
In the first case, all the fields required to complete the wizard for creating a new proposal are empty and must be filled in. While editing a proposal, all the fields of the wizard are pre-filled with its actual data that can be modified to update the proposal.

ADD/EDIT PROPOSAL – PAGE 1

The first page is about selecting the address of the property.

Unfortunately, this is the only page that is completely different from the mockup, because from our research for the implementation of the maps it emerged that it is necessary to make a payment to the Google service in order to implement the Google Maps API.

We have therefore decided to replace it with a simple form that requires the user to enter both the address and the city of the property, without verifying the validity of these two fields for the reason stated above. Note that while the two fields are empty, the button to go to the next step is disabled.

ADD/EDIT PROPOSAL – PAGE 2

In the second page, the user can select the rooms available in his property, a special case is the bedroom, in which once a new bedroom has been added it is possible to specify how many beds are available.

In addition to this, the user can create a new custom room to his proposal.

ADD/EDIT PROPOSAL – PAGE 3

Here the user can set the maximum number of the renters for his facility and, only by doing this, then he is able to add the data of the current renters, if any. In addition, it is possible to edit or remove a current renter.

ADD/EDIT RENTAL PROPOSAL – PAGE 3 (ADDING/EDIT RENTERS PANEL)

In this panel, it is required for the user to fill all the fields to add new renters, or in the editing case, all the fields are pre-filled and the user can modify them.

ADD/EDIT PROPOSAL – PAGE 4

At this step, it is possible to choose all the services provided in the facility by clicking in the entire area of any item in the list.

The user can add a new custom service, then once entered its name, automatically it will be selected, without requiring an additional tap.

ADD/EDIT PROPOSAL – PAGE 5

In this page, the user can set the price for the monthly rent through a slider.

ADD/EDIT PROPOSAL – PAGE 6

In this page the user is asked to fill in the title of the advertisement.
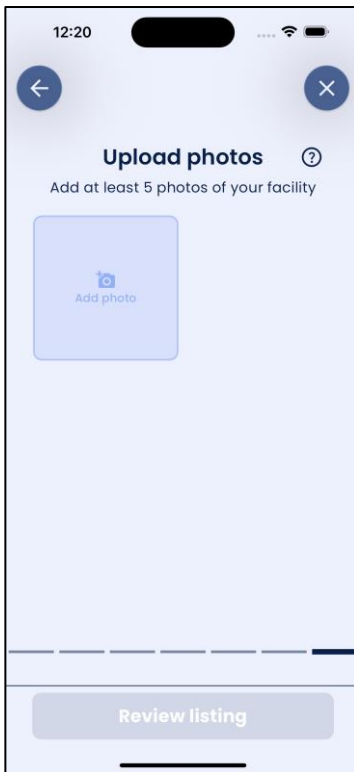
ADD/EDIT PROPOSAL – PAGE 7

Here the user must add at least five photos of the facility by tapping on the "*Add photo*" button, which displays a dialog that makes the user able to choose the source for picking a photo (camera or gallery).

Once the user picked a photo, the image will be displayed as a square in a two-by-two grid list, with the current number of the photo (placed in bottom-right corner of the square) and a delete button (placed in the top-left corner of the square) to remove it.

ADD/EDIT PROPOSAL – PAGE 8

In this final page of the wizard, the user can preview the new proposal created or modified and can decide whether to return to a previous step to modify a specific field or confirm the previously chosen operation.

5.2 Project structure

In this section we describe the project structure that we follow to write the code for RoomFinder. Apart from all the other directories and files created with a new flutter project we want to highlight how we organized the main part of every Flutter project, the "lib/" folder.

```
∨ lib
  > data
  > l10n
  > model
  > presentation
  > provider
  > style
  > util
    firebase_options.dart
    main.dart
```

We decided to use the **layer-first** approach to structure the entire project, a simple structure that allows for separation between the components and ensures a high degree of modularity of the same. As result, every subfolder of "lib/" main directory contains the code for all implemented features.
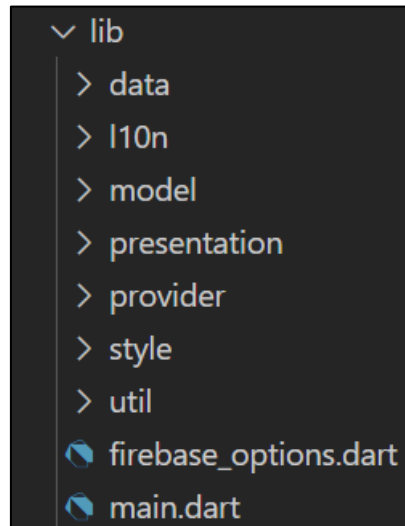
The main subdirectories are:
- **data**: containing the classes definitions to handle data management from the backend;
- **model**: containing the models which represent the data collected from the backend;
- **provider**: containing the Riverpod's providers which assume a role of controller with the aim of redirecting the UI user interaction to the relative data class method which returns as result the data represented by the right model, updating so the UI accordingly with the result of this operation.
- **presentation**: containing the RoomFinder UI.

There are also other subdirectories and files that need to be mentioned:
- **l10n**: containing the constants definitions to handle the app translation in any language. At the moment only the English file has been defined;
- **style**: containing the files to handle the style, both textual and visual, of RoomFinder;
- **util**: containing some useful class to handle particular logic operations that do not belong to data classes, such as the network connection check and the shared_preferences library to store some important variables in the user device;
- **firebase_options.dart**: containing the configuration to access to Firebase services;
- **main.dart**: the main file that runs RoomFinder app.

5.3 External services, frameworks and libraries

This section describes all the external services, frameworks and libraries used to handle effectively and efficiently all the functionalities that RoomFinder provides to users.

5.3.1 Firebase

Firebase is a platform developed by Google that provides a comprehensive set of tools and services for creating web and mobile applications. One of its core components is **Cloud Firestore**, a NoSQL cloud database that we used to handle all the data of RoomFinder application. Data are collected through:

- Collection: a group of related documents.
- Document: an individual record within a collection.
- Subcollection: a collection nested within a document.

The structure that we designed to correctly handle all the functionalities of RoomFinder is following reported. For each collection we report the different fields and subcollections that characterize it.

USERS: a collection that contains all the registered users.
- *id*: unique identifier for the user.
- *isHost*: boolean field indicating if the user is a host or not. Since Firebase Authentication does not provide any roles management, this field is crucial to handle the two types of RoomFinder users.
- *savedAds*: a list of strings containing the reference (uid) to each saved ad. This field is populated only for a user that is a student type.

ADS: a collection that contains all the published ads.
- *id*: unique identifier for the ad.
- *hostID*: reference to the user of type "host" who posted the ad.
- *facilityName*: name of the accommodation.
- *address*: address of the accommodation formed by a street and a city.
    - street: street where the accommodation is located.
    - city: city where the accommodation is located.
- *rooms*: array containing the list of all the facility rooms.
- *rentersCapacity*: maximum number of renters.
- *actualRenters*: array of current renters.
    - name: name of the renter.
    - age: Age of the renter.
    - facultyOfStudies: Faculty of studies of the renter.
    - interests: Interests of the renter.
    - contractDeadline: Deadline for the contract.
- *services*: array of additional services (e.g., Wi-Fi, laundry).
- *monthlyRent*: monthly rent.

This structure allows for efficient storage and retrieval of accommodation listings, user information, and user-saved ads. It also provides flexibility for adding or modifying data fields as needed.

Beyond Cloud Firestore DB, we also used the **Firebase Authentication** service to correctly handle the backend auth workflow and **Firebase Cloud Storage** to store the account and facilities photos.

In particular, since Cloud Storage is an independent service that is not integrated with Cloud Firestore, to correctly retrieve the right photo in the right moment we named every photo (or set of photos) with the users and ads id respectively. This in fact simplifies the data retrieval from backend when needed and handles the relationships between the stored data and photos. To be more precisely, we decided to structure the Firebase Cloud Storage in directories and subdirectories as follows.

photos/
- **users/** (where each profile photo is named with user uid)
- **ads/** (where each subfolder is named with ad uid)
    - **ad_uid/** (each one containing all the photos related to ad uid)

5.3.2 Riverpod

Riverpod is a powerful state management package for Dart and Flutter applications, offering a more robust and flexible alternative to Flutter's native state management solutions. Riverpod facilitates a clear separation of concerns, making it easier to divide the user interface from the business logic. By using Riverpod, we could define and manage state independently of UI components, which enhances the modularity and testability of our code. This separation ensures that the UI remains unaffected by changes in the underlying business logic, leading to cleaner and more maintainable codebases.

5.3.3 flutter_screenutil

Flutter ScreenUtil is a powerful package designed to simplify responsive UI design in Flutter apps. It helps developers create adaptable layouts that automatically adjust to different screen sizes and resolutions. By providing a consistent way to scale dimensions, fonts, and padding, ScreenUtil ensures that RoomFinder maintains a predefined appearance on various devices, from small phones to large tablets. This library is essential for developers who want to achieve pixel-perfect designs without manually handling different screen configurations.

In our case this library was especially significant in switching from initial Figma mockups to Flutter final application. In fact, during mockups design we chose a fixed screen dimension on which draw the app and consequently all the widget dimensions were based on that fixed dimension. What ScreenUtil allowed us to do is setting up the entire app on the same mockups screen dimension and therefore we were able to respect the dimensions chosen during the mobile design step and also writing them in a way that can be scaled following each device screen that runs RoomFinder.

5.3.4 flutter_localization and intl

Both libraries constitute a key package for building multilingual apps. These libraries enable developers to easily translate their apps into different languages respecting plurals, genders, date/number formatting and so on.

We used thes libraries to set up all the textual constants of our application because our goal was to make available even the translation in both English and Italian language but, as we will discuss later on in the "Future implementations" section, we encountered some difficulties in translating the data collected in Cloud Firestore and for this we were not able to implement the translation. Anyway, we decided to keep the done work with both libraries because we think that having all the constants in just one file, a key characteristic of Flutter Localization, can improve the code maintainability and even can be a set up for a future translation implementation.

5.3.5 shared_preferences

Shared Preferences is a popular Flutter package used for storing simple data persistently on the user's device. Shared Preferences is ideal for storing lightweight data without the need for a complex database, making it a go-to solution for developers looking to implement quick and efficient data persistence in their apps.

We used this library to handle the welcome message page, in a way that once the users have seen it, that page does not appear anymore. In fact, the idea of that page is presenting RoomFinder application to new users only.

5.3.6 path_provider and http

path_provider is a Flutter plugin for finding commonly used locations on the filesystem. Supports Android, iOS, Linux, macOS and Windows while http package contains a set of high-level functions and classes that make it easy to consume HTTP resources. It is multi-platform (mobile, desktop, and browser) and supports multiple implementations.

These two libraries are used in the step seven of the wizard, in which a user can choose the photos for his facility, but only in the editing mode, in order to display the actual photos of the proposal created previously. In detail, starting from a photo URL retrieved from Cloud Firestorage, and then given it in input to http package, it downloads for us the photo and then, in combination with path_provider, makes us able to create a File object needed to display a photo.

5.4 Testing

The final application has been tested over several virtual devices, including both iOS and Android platforms and obtaining the same successful results on both platform as the cross-platform development should do. RoomFinder has been tested also on respectively one Android and one iOS real device revealing the successful result in what developed in terms of look&feel and how the app works.

5.5 Future implementations

As previously discussed for the chat page, there were some initial ideas, not implemented due to the lack of time, that could power up RoomFinder application. In this section we report those ideas that can be considered as future implementations to enhance the RoomFinder user experience.

The future implementations are following reported:

- **Search bar autocompletion**: a standard way to help the user when he interacts with a search functionality is to provide an autocompletion for the results that match with the digited content. Now, due unfortunately to time constraints, this behaviour is not handled and the user needs to digit the right city to find concrete results, if any.

- **App translation**, both in Italian and English: this was a planned feature that we didn't implement, again due to the lack of time. In fact, the app is currently structured to support the Italian and English translation but just for the app textual elements that does not depend on the retrieved data. In fact, the main challenge was to find some way to translate the data collected in Firebase DB that would have required more time to analyse possible solutions.

- **Chat modality**: this was something that we would have liked to implement because is a functionality always offered by the competitors' applications that allows a student to request precious information about specific rental proposals directly to the host responsible for that facility.